WEST

Generate Collection | Print

Search Results - Record(s) 1 through 1 of 1 returned.

☐ 1. Document ID: NN86034506

L19: Entry 1 of 1 File: TDBD Mar 1, 1986

TDB-ACC-NO: NN86034506

DISCLOSURE TITLE: Register Allocation

PUBLICATION-DATA:
IBM Technical Disclosure Bulletin, March 1986, US

VOLUME NUMBER: 28
ISSUE NUMBER: 10
PAGE NUMBER: 4506 - 4513

DISCLOSURE TEXT:

- An algorithm is disclosed that allocates machine registers for IBM System/370 architecture. The above figure depicts a flow chart for an improved register allocator, the major task of which is to assign unlimited "symbolic registers" (SR) to the finite register set on the hardware. Following is a summary of each stage in the process: 1. Right Number of Names (RNN). A SR may have several disjoint lifetimes, i.e., different sequences of instructions in which it can reside in distinct real registers. RNN calculates the lifetimes of all SRs, and separates the disjoint lifetimes of each SR into different "names". Each of these names is actually assigned to a register. 2. Reduce Register Pressure (RRP) and RX Rematerialization. The register pressure at an instruction is the number of names alive at that instruction. Since each name needs a real register, the register pressure at every point in the program must be less than or equal to the number of available hardware registers for register allocation to succeed. RRP reduces register pressure at points where it is too high by "spilling", i.e., storing some live values into memory, and later reloading them. Earlier compiler phases express all operations in terms of register-to-register (RR) instructions; this is most efficient when an SR is used several times. However, if the SR is only used once, it is often possible to employ a single memory to register (RX) instruction to achieve the same effect as the two instruction sequence of: Load, RR instruction. RX Rematerialization converts such sequences into the equivalent RX instruction where possible. 3. Right Number of Names (RNN). RNN is run again after spilling because the STORE/LOAD pairs break single lifetimes into several lifetimes, and RX rematerialization eliminates the need for some registers altogether, so RNN yields a result different from the first time. More important, it is easier to allocate registers for the resulting "names". 4. Build Interference Graph (BIG). Two names are said to interfere if their lifetimes overlap. The interference relation is represented by the interference graph (IG), which is an undirected graph with names as nodes, and edges connecting interfering names. A coloring of a graph is an assignment of integers (real registers) to nodes (names) such that adjacent nodes (interfering names) are assigned different integers (real registers). Register allocation is equivalent to coloring the interference graph with real register colors. BIG constructs the interference graph by examining each instruction and adding edges between all names which are live at that point. The IG is also exploited to express some of the irregularities of the hardware. For instance, to prevent register zero (R0) from being used as a base or index register, R0 is made to interfere with every name appearing as a base or index in some instruction. This process is also accomplished during the scan of the program. 5. Coalesce. Earlier phases of the compiler insert (explicitly and implicitly) numerous load register instructions: LR Ra, Rb If Ra and Rb are mapped to the same real register, then the LR can be eliminated. The purpose of Coalesce is to cause as many LRs to be eliminated by this method as possible. Coalesce does this by scanning the program for LRs. If the names of the two symbolic registers do not interfere, the names are combined into a single name and their interferences are joined. In terms of the IG, one of the nodes is chosen as the representative, and the other's edges are added to it. This guarantees that Color will assign the original two names to the same register. 6. Color. This colors the IG (as modified by Coalesce) with a real register number of colors to provide actual mapping of names to real registers. If the coloring attempt is unsuccessful, Fixup is invoked to fix those names that remain uncolored. 7. Fixup. Fixup is executed if any names remain uncolored after Color. Code is inserted to cause these names to be spilled everywhere except at the individual instructions where mentioned. 8. Spill Temp Allocation. Just as names must be assigned to real registers, so too must the spill temps be assigned to memory locations over their lifetimes. The problem is similar to register allocation. Spill temps having non-overlapping lifetimes can share the same storage location. The major difference is that there are an unlimited number of memory locations, instead of a fixed set of registers. Spill temp allocation uses a modified version of BIG and Color for its processing. The structure of the improved register allocator is more efficient than that of the prior art. The improved register allocator makes fewer passes over the program; register pressure is always reduced before attempting to Color, since it is almost always necessary to spill something; the algorithm never loops - if there are remaining uncolored names, they are spilled by Fixup; and the IG data structure is represented in a space-efficient manner, at a minor decrease in time efficiency. The IG consists solely of the adjacency lists (halved). There are two instances where the prior-art bit matrix is advantageous. 1. In BIG, duplicates are kept out of the adjacency list by testing matrix (i,j). 2. In Coalesce, for each LF i,j , Matrix (i,j) is tested to see if i and j interfere. In the improved allocator, the adjacency lists are used alone in these two situations, and are not inefficient because: 1. In BIG, duplicates are not forbidden. So BIG simply adds j to i's adjacency list without checking for a duplicate. This potentially uses too much space, so if a space limit is reached, BIG performs a "garbage collection". The process uses a single temporary bit vector, initialized to all zeros before using on each list. Each adjacency list is scanned: if the corresponding bit is '1', the entry is a duplicate and is reclaimed; otherwise, the bit is set to '1'. In practice, few duplicates are generated, because edges are added at the definition point of a name, and most names have a single definition point. 2. During Coalesce, it is necessary to decide if i and j interfere. Thus it is necessary to traverse i's list looking for j, which might be inefficient if i's list is long. But, there are several reasons why this inefficiency does not arise. a. Garbage collection is performed at the beginning of Coalesce, so the lists are as short as possible. b. Since RRP is run before Coalesce, register pressure has been reduced to a fixed maximum, say, 14. This implies that most names interfere with less than 14 other names; i.e., the average number of interferences for a name is independent of the total number of names. c. The lists are halved, so the average list length is half of the number of adjacent elements, e.g., 6.5. d. If i and j do interfere, then on average j will be found after scanning only half of i's list, e.g., 3.25. e. If i and j do not interfere, then i's entire list will be scanned. In this case, i and j will be coalesced, but the Coalesce action requires scanning i's list anyway. Thus, at worst, it costs twice as much, but the cost is not proportional to the length of i's list. The improved algorithm deals with 16 general registers and 4 floating point registers, whereas the prior-art algorithm only handles the general registers. The floating-point registers are independent of the general registers, i.e., there are no instructions using registers from the two register classes. Therefore, the improved process identifies symbolic registers by class, and the register allocation process is applied to the two classes independently. For efficiency, the two classes are treated in parallel during a single execution of the steps in the figure. This permits fewer passes over the program. The improved algorithm deals with register pairs, whereas the prior-art process does not. Certain instructions on the IBM System/370 use even-odd pairs of general registers: a. Multiply: M, MR b. Divide: D, DR c. Move Long MVCL, and Compare Long CLCL d. Branch on Index High, BXH, and Branch on Index Low or Equal BXLE. Register pairs are more difficult to allocate optimally than floating-point registers, because the pairs overlap with the single registers. The mechanism is as follows: 1. The SRs are classified as single or pair in the intermediate language. 2. There are intermediate language instructions that convert singles to/from pairs: - Combine: generates a pair from two singles. This operation normally expands into two Load Register (LR) hardware instructions, to move the two single registers into the corresponding members of the pair. The instructions in classes c and d above have pair registers as input, which are generated by Combine. - Extract_Odd, Extract_Even: generates a single from a pair by extracting the odd or even number of the pair, respectively. This operation normally expands into the machine instruction Load Register (LR), to move the value from a member of the pair to the single register. The instructions in classes a and b above create pair results from which single values are obtained using Extract. 3. RNN. Pairs are treated like singles. 4. RRP. Register pressure is computed for the general registers counting 2 for pairs and 1 for singles. RX Rematerialization treats pairs like singles. 5. BIG. Pairs are treated like singles: pairs can interfere with singles, and vice versa. 6. Coalesce. There are two unique processes of performing coalesce with pairs: A. In one process, pairs only participate in Coalesce when they appear in Load Pair Register instructions, which copy from one pair register to another. Pairs do not participate in any other way during Coalesce. B. The second and

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments

KWIC | Draw Desc | Clip Img

Generate Collection | Print

Terms	Documents
L15	1

Display Format: REV | Change Format

Previous Page Next Page

WEST Search History

DATE: Tuesday, October 28, 2003

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
	DB=USPT,PGPB; PLUR=YES; OP=ADJ		
L12	L11 and index$	3	L12
L11	L6 and (scratch near register)	9	L11
	DB=JPAB; PLUR=YES; OP=ADJ		
L10	L6	0	L10
	DB=EPAB,DWPI; PLUR=YES; OP=ADJ		
L9	L6	1	L9
	DB=TDBD; PLUR=YES; OP=ADJ		
L8	L6	0	L8
	DB=USPT,PGPB; PLUR=YES; OP=ADJ		
L7	L6	88	L7
	DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
L6	L4 and ((index$ near2 register) or scratch)	89	L6
L5	L4 and (index$ near2 register)	68	L5
L4	L3 and (allocat$ near register)	357	L4
L3	instrument$ or profil$	1074356	L3
	DB=USPT,PGPB; PLUR=YES; OP=ADJ		
L2	((714/34 |714/35 |714/36 |714/37 |714/38 |714/39)!.CCLS.)	1709	L2
L1	((717/124 |717/127 |717/128 |717/129 |717/130 |717/131 |717/132 |717/133 |717/141 |717/142 |717/143 |717/144 |717/145 |717/155 |717/156 |717/158 |717/159)!.CCLS.)	1735	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 9 of 9 returned.

☐ 1. Document ID: US 20030126591 A1

L11: Entry 1 of 9 File: PGPB Jul 3, 2003

PGPUB-DOCUMENT-NUMBER: 20030126591
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030126591 A1

TITLE: Stride-profile guided prefetching for irregular code

PUBLICATION-DATE: July 3, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Wu, Youfeng	Palo Alto	CA	US	
Serrano, Mauricio	San Jose	CA	US	

US-CL-CURRENT: 717/158; 717/159

ABSTRACT:

A compiler technique uses profile feedback to determine stride values for memory references, allowing prefetching of instructions for those loads that can be effectively prefetched. The compiler first identifies a set of loads, and instruments the loads to profile the difference between the successive load addresses in the current iteration and in the previous iteration. The frequency of stride difference is also profiled to allow the compiler to insert prefetching instructions for loads with near-constant strides. The compiler employs code analysis to determine the best prefetching distance, to reduce the profiling cost, and to reduce the prefetching overhead.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 2. Document ID: US 20020199179 A1

L11: Entry 2 of 9 File: PGPB Dec 26, 2002

PGPUB-DOCUMENT-NUMBER: 20020199179
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020199179 A1

TITLE: Method and apparatus for compiler-generated triggering of auxiliary codes

PUBLICATION-DATE: December 26, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Lavery, Daniel M.	Santa Clara	CA	US	
Wang, Hong	Fremont	CA	US	
Hoflehner, Gerolf F.	Santa Clara	CA	US	
Liao, Shih-wei	Sunnyvale	CA	US	
Shen, John	San Jose	CA	US	
Grochowski, Edward T.	San Jose	CA	US	
Sehr, David C.	Sunnyvale	CA	US	
Fang, Jesse Z.	San Jose	CA	US	

US-CL-CURRENT: 717/158

ABSTRACT:

A method for executing a code is provided. The method includes receiving a trigger instruction, selecting an entry in a trigger table, the entry associated with the trigger instruction, and executing an auxiliary code referenced by the entry in the trigger table.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 3. Document ID: US 20020144245 A1

L11: Entry 3 of 9 File: PGPB Oct 3, 2002

PGPUB-DOCUMENT-NUMBER: 20020144245
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020144245 A1

TITLE: Static compilation of instrumentation code for debugging support

PUBLICATION-DATE: October 3, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Lueh, Guei-Yuan	San Jose	CA	US	

US-CL-CURRENT: 717/140

ABSTRACT:

The present invention is a method and system to support debug. A function is compiled. The function includes a byte code sequence having a field byte code that accesses or modifies a field. The compiled function provides a native code and occupies a code space. An instrumentation code corresponding to a field match of a field is generated. The instrumentation code is inserted to the native code.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 4. Document ID: US 20020144241 A1

L11: Entry 4 of 9 File: PGPB Oct 3, 2002

PGPUB-DOCUMENT-NUMBER: 20020144241
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020144241 A1

TITLE: Debugging support using dynamic re-compilation

PUBLICATION-DATE: October 3, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Lueh, Guei-Yuan	San Jose	CA	US	

US-CL-CURRENT: 717/136

ABSTRACT:

The present invention is a method and system to support debug. A function is re-compiled when a field watch for a field is activated. The function includes a byte code sequence having a field byte code that accesses or modifies the field. The re-compiled function provides a native code and occupies a code space. An instrumentation code corresponding to the field watch of the field is generated. The instrumentation code is inserted to the native code.

☐ 5. Document ID: US 20020083425 A1

L11: Entry 5 of 9   File: PGPB   Jun 27, 2002

PGPUB-DOCUMENT-NUMBER: 20020083425
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020083425 A1

TITLE: System and method for obtaining scratch registers in computer executable binaries

PUBLICATION-DATE: June 27, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Gillies, David M.	Bellevue	WA	US	
Chaiken, Ronnie	Woodinville	WA	US	
Liu, Jiyang	Sammamish	WA	US	

US-CL-CURRENT: 717/168

ABSTRACT:

A system and method for obtaining scratch registers in a computer-executable binary is provided. Register allocation requests in a computer-executable binary are discovered. In one method, the register allocations are examined procedure-by-procedure. The maximum number of registers requested by any instruction in the procedure is discovered. Then, register requests in the procedure are modified to request the maximum number discovered plus a number of scratch registers. In another method, the register allocations are examined block-by block within a procedure. Dominating register allocations for each block are found. Then the dominating register allocations are modified to request scratch registers.

☐ 6. Document ID: US 6631514 B1

US-PAT-NO: 6631514
DOCUMENT-IDENTIFIER: US 6631514 B1

TITLE: Emulation system that uses dynamic binary translation and permits the safe speculation of trapping operations

DATE-ISSUED: October 7, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Le; Bich-Cau	San Jose	CA		

US-CL-CURRENT: 717/137; 717/136, 717/151, 717/159

ABSTRACT:

The inventive emulator dynamically translates instructions in code written for a first architecture into code for a second architecture. The emulator designates various checkpoints in the original code, and speculatively reorders the placement of the translated code instructions according to optimization procedures. If during the execution of the reordered code, a trap should occur, then the emulator resets the original code to the most recent checkpoint and begins executing the original code sequentially in a line-by-line manner until the section is completed or branched out of. The original code is reset by changing the program counter to the checkpoint, and reversing the effects of each instruction which has been executed subsequent to the checkpoint. Thus, any native instructions which correspond to original instructions which occur sequentially prior to the checkpoint have been executed, and any native instructions which correspond to original instructions which occur sequentially subsequent to the checkpoint have not been executed.

52 Claims, 9 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

---

☐ 7. Document ID: US 6625807 B1

L11: Entry 7 of 9 File: USPT Sep 23, 2003

US-PAT-NO: 6625807
DOCUMENT-IDENTIFIER: US 6625807 B1

TITLE: Apparatus and method for efficiently obtaining and utilizing register usage information during software binary translation

DATE-ISSUED: September 23, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chen; Ding-Kai	San Jose	CA		

US-CL-CURRENT: 717/154; 703/23, 712/227, 717/128, 717/136

ABSTRACT:

Apparatus and method are described for register optimization during code translation and utilizes a technique that removes the time overhead for analyzing register usage, and eliminates fixed restraints on the compiler register usage. The present invention for register optimization utilizes a compiler to produce a bit vector for each program unit (i.e., subroutine, function, and/or procedure). Each bit in the bit vector represents a particular caller-saved register. A bit is set if the compiler uses the corresponding register within that program unit. During the translation, the translator examines the bit vector to very quickly determine which registers are free, and therefore can be used during register optimization without having to save and restore the register values.

16 Claims, 12 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 10

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

☐ 8. Document ID: US 6434620 B1

L11: Entry 8 of 9   File: USPT   Aug 13, 2002

US-PAT-NO: 6434620
DOCUMENT-IDENTIFIER: US 6434620 B1

TITLE: TCP/IP offload network interface device

DATE-ISSUED: August 13, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Boucher; Laurence B.	Saratoga	CA		
Blightman; Stephen E. J.	San Jose	CA		
Craft; Peter K.	San Francisco	CA		
Higgen; David A.	Saratoga	CA		
Philbrick; Clive M.	San Jose	CA		
Starr; Daryl D.	Milpitas	CA		

US-CL-CURRENT: 709/230; 709/250

ABSTRACT:

An intelligent network interface card (INIC) or communication processing device (CPD) works with a host computer for data communication. The device provides a fast-path that avoids protocol processing for most messages, greatly accelerating data transfer and offloading time-intensive processing tasks from the host CPU. The host retains a fallback processing capability for messages that do not fit fast-path criteria, with the device providing assistance such as validation even for slow-path messages, and messages being selected for either fast-path or slow-path processing. A context for a connection is defined that allows the device to move data, free of headers, directly to or from a destination or source in the host. The context can be passed back to the host for message processing by the host. The device contains specialized hardware circuits that are much faster at their specific tasks than a general purpose CPU. A preferred embodiment includes a trio of pipelined processors devoted to transmit, receive and utility processing, providing full duplex communication for four Fast Ethernet nodes.

6 Claims, 59 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 40

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

☐ 9. Document ID: US 6427234 B1

L11: Entry 9 of 9 File: USPT Jul 30, 2002

US-PAT-NO: 6427234
DOCUMENT-IDENTIFIER: US 6427234 B1
** See image for Certificate of Correction **

TITLE: System and method for performing selective dynamic compilation using run-time information

DATE-ISSUED: July 30, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chambers; Craig	Seattle	WA		
Eggers; Susan J.	Seattle	WA		
Grant; Brian K.	Shoreline	WA		
Mock; Markus	Seattle	WA		
Philipose; Matthai	Seattle	WA		

US-CL-CURRENT: 717/140; 717/148, 717/153

ABSTRACT:

Selective dynamic compilation of source code is performed using run-time information. A system is disclosed that implements a declarative, annotation based dynamic compilation of the source code, employing a partial evaluation, binding-time analysis (BTA), and including program-point-specific polyvariant division and specialization and dynamic versions of traditional global and peephole optimizations. The system allows programmers to declaratively specify policies that govern the aggressiveness of specialization and caching, providing fine control over the dynamic compilation process. The policies include directions for controlling specialization at promotion points and merge points, and further define caching policies, and speculative-specialization policies. The system also enables programmers to specialize programs across arbitrary edges, both at traditional locations, such as procedure boundaries, but also within procedures. Programmers are enabled to conditionally specialize programs based on evaluation of arbitrary compile-time and run-time conditions.

20 Claims, 32 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 22

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

Generate Collection | Print

Terms	Documents
L6 and (scratch near register)	9

Display Format: - Change Format

Previous Page Next Page

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 1 of 1 returned.

☐ 1. Document ID: US 20020083425 A1

```
L9: Entry 1 of 1                File: DWPI              Jun 27, 2002
```

DERWENT-ACC-NO: 2002-590185
DERWENT-WEEK: 200263
COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Computer-implemented method for instrumenting binaries, involves modifying each register request from several register requests to request maximum number of discovered registers plus scratch registers

INVENTOR: CHAIKEN, R; GILLIES, D M ; LIU, J

PRIORITY-DATA: 2000US-0746949 (December 21, 2000)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 20020083425 A1	June 27, 2002		018	G06F009/44

INT-CL (IPC): G06 F 9/44

ABSTRACTED-PUB-NO: US20020083425A
BASIC-ABSTRACT:

NOVELTY - The maximum number of registers requested from several register requests are determined. Each register request is modified to request the maximum number of registers with selected number of scratch registers.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are included for the following:

(1) Computer system; and

(2) Computer readable medium.

USE - For instrumenting binaries for hardware architecture in computer (claimed).

ADVANTAGE - A linear search for maximum register allocation is performed on a procedure-by-procedure basis, and the linear replacement is performed to provide scratch registers effectively without requiring the involved analysis. The scratch registers are provided using the same index throughout the procedure, thus simplifies the inserting of instrumenting code effectively.

Search Results - Record(s) 1 through 7 of 7 returned.

1. Document ID: US 6195793 B1

```
L13: Entry 1 of 7                    File: USPT                    Feb 27, 2001
```

US-PAT-NO: 6195793
DOCUMENT-IDENTIFIER: US 6195793 B1

TITLE: Method and computer program product for adaptive inlining in a computer system

DATE-ISSUED: February 27, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Schmidt; William Jon	Rochester	MN		

US-CL-CURRENT: 717/151; 717/157

ABSTRACT:

A method and computer program product are provided for implementing adaptive inlining in a computer system. Call sites in a call multigraph are identified for possible inlining. A first approximation of initial call sites of the identified possible call sites are identified for inlining. Procedures in the call multigraph are processed in a determined order where a first procedure is only processed after all second procedures called by the first procedure are processed. The processing of the first procedure comprises the steps of determining whether any call site within the first procedure has been selected for inlining, and whether the second procedure called from the call site contains confirmed inlined call sites. If true, it is determined whether to confirm or reject the first approximation to inline the second procedure into the first procedure at the call site utilizing at least one predetermined criterion.

18 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

2. Document ID: US 6175956 B1

```
L13: Entry 2 of 7                    File: USPT                    Jan 16, 2001
```

US-PAT-NO: 6175956
DOCUMENT-IDENTIFIER: US 6175956 B1

TITLE: Method and computer program product for implementing method calls in a computer system

DATE-ISSUED: January 16, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hicks; Daniel Rodman	Byron	MN		
Schmidt; William Jon	Rochester	MN		

US-CL-CURRENT: 717/114; 717/157; 717/158

ABSTRACT:

A computer implemented method and computer program compiler product are provided for implementing method calls in a computer system. Virtual method calls are identified in an intermediate instruction stream representation. Responsive to an identified virtual method call, profile data for the identified call site are read. A most frequently called procedure for the identified call site is compared with a first threshold value. Responsive to the most frequently called procedure being called less than the first threshold value, the virtual method call is maintained in a revised instruction stream representation. Responsive to the most frequently called procedure being called greater than or equal to the first threshold value, a guarded call to the most frequently called procedure is inserted at the identified call site in the revised instruction stream representation. In accordance with features of the invention, checking whether one object type accounts for more than a second threshold value of the calls to the most frequently called procedure at the identified call site is performed. Responsive to one object type accounting for more than or equal to the second threshold value, a type guard and a call to the most frequently called procedure are inserted at the identified call site in the revised instruction stream representation. Responsive to one object type accounting for less than the second threshold value, an address guard and a call to the most frequently called procedure are inserted at the identified call site in the revised instruction stream representation.

13 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

---

☐ 3. Document ID: US 6072951 A

L13: Entry 3 of 7   File: USPT   Jun 6, 2000

US-PAT-NO: 6072951
DOCUMENT-IDENTIFIER: US 6072951 A

TITLE: Profile driven optimization of frequently executed paths with inlining of code fragment (one or more lines of code from a child procedure to a parent procedure)

DATE-ISSUED: June 6, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Donovan; Robert John	Rochester	MN		
Roediger; Robert Ralph	Rochester	MN		
Schmidt; William Jon	Rochester	MN		

US-CL-CURRENT: 717/158; 717/159

ABSTRACT:

A compiler and method of compiling provide enhanced performance by inlining one or more frequently executed paths through a child procedure into a parent procedure without inlining the entire child procedure. Accordingly, a substantial improvement in speed of execution of the program can be achieved by reducing procedure call

overhead, with reduced expense in terms of program size as compared to traditional inlining. Various criteria for determining whether to inline particular child procedures are also described.

26 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

---

☐ 4. Document ID: US 6029000 A

L13: Entry 4 of 7 File: USPT Feb 22, 2000

US-PAT-NO: 6029000
DOCUMENT-IDENTIFIER: US 6029000 A

TITLE: Mobile communication system with cross compiler and cross linker

DATE-ISSUED: February 22, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Woolsey; Matthew A.	Plano	TX		
Lineberry; Marion C.	Dallas	TX		
Kim; Jihong	Taegu			KR

US-CL-CURRENT: 717/147; 717/114, 717/162, 717/173

ABSTRACT:

A wireless data platform (10) comprises a plurality of processors (12,16). Channels of communication are set up between processors such that they may communicate information as tasks are performed. A dynamic cross compiler (80) executed on one processor compiles code into native processing code for another processor. A dynamic cross linker (82) links the compiled code for other processor. Native code may also be downloaded to the platform through use of a JAVA Bean (90) (or other language type) which encapsulates the native code. The JAVA Bean can be encrypted and digitally signed for security purposes.

17 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

---

☐ 5. Document ID: US 6016398 A

L13: Entry 5 of 7 File: USPT Jan 18, 2000

US-PAT-NO: 6016398
DOCUMENT-IDENTIFIER: US 6016398 A

TITLE: Method for using static single assignment to color out artificial register dependencies

DATE-ISSUED: January 18, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Radigan; Jim J.	San Jose	CA		

US-CL-CURRENT: 717/152

ABSTRACT:

The invention is a method of using static single assignment intermediate language to color out artificial register dependencies while compiling at least a portion of a computer program. The method comprises creating a rank-n SSA intermediate language representation of the computer program, wherein n is a positive integer greater than 1; and coloring out the artificial register dependencies.

22 Claims, 13 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 12

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

---

☐ 6. Document ID: US 5937195 A

L13: Entry 6 of 7   File: USPT   Aug 10, 1999

US-PAT-NO: 5937195
DOCUMENT-IDENTIFIER: US 5937195 A

TITLE: Global control flow treatment of predicated code

DATE-ISSUED: August 10, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Ju; Dz-ching	Sunnyvale	CA		
Gillies; David M.	Cupertino	CA		

US-CL-CURRENT: 717/156; 712/201, 712/216, 717/159

ABSTRACT:

The relationships among predicates are tracked globally by uniformly treating both control flow and explicit predicates by mapping them to a single connected partition graph. This allows for the analysis of predicate relations based on the scope of an entire procedure. This predicate analysis can be invoked by various phases of compiler optimization without being constrained by an incremental update of any persistent data structures.

40 Claims, 8 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

---

☐ 7. Document ID: US 5768595 A

L13: Entry 7 of 7   File: USPT   Jun 16, 1998

US-PAT-NO: 5768595
DOCUMENT-IDENTIFIER: US 5768595 A
** See image for Certificate of Correction **

TITLE: System and method for recompiling computer programs for enhanced optimization

DATE-ISSUED: June 16, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Gillies; David M.	Ontario			CA

US-CL-CURRENT: 717/145; 717/156, 717/159

ABSTRACT:

An optimizing compiler for producing executable programs from code, high level languages compiles the code whilst generating data from which a callgraph may be constructed, and then recompiles the procedures identified in the callgraph in an order which reverses the topology of the callgraph while monitoring usage of hardware registers. Procedures which are rarely or never called, or result in termination of the program, are identified, and are modified if needed so that if called, registers which they may modify are saved prior to execution of the procedure and subsequently restored if necessary, so that in a calling procedure, subsequently recompiled, no account need be taken of possible register usage by the called procedure. This makes additional registers available to the calling procedure, and enables register storing and restoring which must otherwise be associated with the callsite to be eliminated.

16 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments   KWIC | Draw Desc | Image

Generate Collection   Print

Terms	Documents
5768595	7

Display Format: -   Change Format

Previous Page   Next Page

PORTAL THE ACM DIGITAL LIBR

> home > about > feedback > login
US Patent & Trademark Office

Try the *new* Portal design
Give us your opinion after using it.

Search Results

Search Results for: **[instrument<AND>((indexing<AND>((scratch register))))]**
Found **10** of **121,820 searched.**

Search within Results

> Advanced Search
> Search Help/Tips

Sort by: Title Publication Publication Date Score Binder

Results 1 - 10 of 10 short listing

1 An environment for research in microprogramming and emulation 80%
Robert F. Rosin , Gideon Frieder , Richard H. Eckhouse
Communications of the ACM August 1972
Volume 15 Issue 8
The development of the research project in microprogramming and emulation at State University of New York at Buffalo consisted of three phases: the evaluation of various possible machines to support this research; the decision to purchase one such machine, which appears to be superior to the others considered; and the organization and definition of goals for each group in the project. Each of these phases is reported, with emphasis placed on the early results achieved in this research.

2 System architectures for computer music 80%
John W. Gordon
ACM Computing Surveys (CSUR) June 1985
Volume 17 Issue 2
Computer music is a relatively new field. While a large proportion of the public is aware of computer music in one form or another, there seems to be a need for a better understanding of its capabilities and limitations in terms of synthesis, performance, and recording hardware. This article addresses that need by surveying and discussing the architecture of existing computer music systems. System requirements vary according to what the system will be used for. Common uses for co ...

3 Trap architectures for Lisp systems 77%
Douglas Johnson
Proceedings of the 1990 ACM conference on LISP and functional programming May 1990
Recent measurements of Lisp systems show a dramatic skewing of operation frequency. For example, small integer (fix-num) arithmetic dominates most programs, but other number

types can occur on almost any operation. Likewise, few memory references trigger special handling for garbage collection, but nearly all memory operations could trigger such special handling. Systems like SPARC and SPUR have shown that small amounts of special hardware can significantly reduce the need for inline softwa ...

4 Construction of a transportable, multi-pass compiler for extended Pascal 77%
G. J. Hansen , G. A. Shoults , J. D. Cointment
Proceedings of the 1979 SIGPLAN symposium on Compiler construction August 1979
This paper describes the implementation of an extended Pascal compiler on the TI 990 minicomputer, the TI 980 minicomputer, and the IBM System 370. The compiler was designed to be as machine independent as possible; the parser and machine independent optimizer are parameterized so they are identical at the source code level for all machines. The paper describes the language modifications and extensions to Pascal, the parser, the optimizer, and the code generators. In addition, the intermedi ...

5 Performance monitoring: METRIC: tracking down inefficiencies in the memory hierarchy via binary rewriting 77%
Jaydeep Marathe , Frank Mueller , Tushar Mohan , Bronis R. de Supinski , Sally A. McKee , Andy Yoo
In this paper, we present METRIC, an environment for determining memory inefficiencies by examining data traces. METRIC is designed to alter the performance behavior of applications that are mostly constrained by their latency to resolve memory references. We make four primary contributions in this paper. First, we present methods to extract partial data traces from running applications by observing their memory behavior via dynamic binary rewriting. Second, we present a methodology to represent ...

6 Dynamo: a transparent dynamic optimization system 77%
Vasanth Bala , Evelyn Duesterwald , Sanjeev Banerjia
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation May 2000
Volume 35 Issue 5
We describe the design and implementation of Dynamo, a software dynamic optimization system that is capable of transparently improving the performance of a native instruction stream as it executes on the processor. The input native instruction stream to Dynamo can be dynamically generated (by a JIT for example), or it can come from the execution of a statically compiled native binary. This paper evaluates the Dynamo system in the latter, more challenging situation, in order to emphasize the ...

7 Direct execution models of processor behavior and performance 77%
Richard M. Fujimoto , William B. Campbell
Proceedings of the 19th conference on Winter simulation December 1987
This paper discusses a modeling technique for creating efficient instruction level simulation models of von Neumann processors. In contrast to traditional approaches which use a software interpreter, this technique employs direct execution of application programs on the host computer. An assembly language program for the target machine is decompiled to a high level language, instrumented, and then recompiled and executed on the host computer. A prototype im ...

8 Trap-driven memory simulation with Tapeworm II 77%
Richard Uhlig , David Nagle , Trevor Mudge , Stuart Sechrest
ACM Transactions on Modeling and Computer Simulation (TOMACS) January 1997
Volume 7 Issue 1

9 Shade: a fast instruction-set simulator for execution profiling 77%
Bob Cmelik , David Keppel
ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems May 1994
Volume 22 Issue 1
Tracing tools are used widely to help analyze, design, and tune both hardware and software systems. This paper describes a tool called Shade which combines efficient instruction-set simulation with a flexible, extensible trace generation capability. Efficiency is achieved by dynamically compiling and caching code to simulate and trace the application program. The user may control the extent of tracing in a variety of ways; arbitrarily detailed application state information may be collected ...

10 Fast instruction cache performance evaluation using compile-time analysis 77%
David B. Whalley
ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1992 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems June 1992
Volume 20 Issue 1

Results 1 - 10 of 10 short listing

CiteSeer Find: scratch register allocate indexing Documents Citations

Searching for **PHRASE scratch register allocate indexing**.
Restrict to: Header Title Order by: Citations Hubs Usage Date Try: Amazon B&N Google (RI) Google (Web) CSB DBLP
No documents match Boolean query. Trying non-Boolean relevance query.
1000 documents found. **Only retrieving 250 documents (System busy - maximum reduced).** Retrieving documents... **Order: relevance to query.**

Fast Context Switches: Compiler and Architectural Support .. - Snyder, Whalley, Baker (Correct) (1 citation)
on a live range that has been **allocated** to a **scratch register**. 2 The **scratch register** can be
this paper attempts to avoid saving and restoring **registers** by performing context switches at points in the
www.cs.fsu.edu/~whalley/papers/mnm95.ps

A New Fast Algorithm for Optimal Register Allocation in.. - Lelait, Gao, Eisenbeis (1998) (Correct) (1 citation)
En Automatique A New Fast Algorithm For Optimal **Register** Allocation In Modulo Scheduled Loops Sylvain
ftp.inria.fr/INRIA/publication/RR/RR-3337.ps.gz

Global Register Allocation Based on Graph Fusion - Guei-Yuan Lueh (1996) (Correct) (7 citations)
Global **Register** Allocation Based on Graph Fusion Guei-Yuan Lueh
www.cs.cmu.edu/afs/cs.cmu.edu/project/iwarp/archive/fx-papers/lcpc96.ps

Register Allocation over the Program Dependence Graph - Norris, Pollock (1994) (Correct) (19 citations)
Register Allocation over the Program Dependence Graph
www.eecis.udel.edu/pub/people/pollock/rap.ps

Code Reordering and Speculation Support for Dynamic.. - Nystrom, Barnes.. (2001) (Correct)
ordering of exceptions and the observed processor **register** contents at each exception point must be
live out of branch I. Compilers may also **register allocate** across exception paths allowing **registers** to be
the counter will contain the appropriate **index** into the table. The processor discards any
www.crhc.uiuc.edu/IMPACT/ftp/conference/pact-01-speculation.ps

Software-Directed Register Deallocation for.. - Lo, Parekh, Eggers, .. (Correct) (2 citations)
and Distributed Systems Software-Directed **Register** Deallocation for Simultaneous Multithreaded
www-cse.ucsd.edu/users/tullsen/TPDS99.ps

Dynamic Statistics of Sequential Prolog - Celis, Mills (Correct)
the architecture has 8 argument **registers** and no **scratch registers**. Hence, for each procedure calls that
is done by updating the current choice pointer **register** (B) with the value of the previous choice
and trust_me_else. Every time an environment is **allocated**, the flag and the current choice point are
ftp.cs.indiana.edu/pub/techreports/TR390.ps.Z

Register Relocation: Flexible Contexts for Multithreading - Waldspurger, Weihl (1993) (Correct) (25 citations)
Register Relocation: Flexible Contexts for
www.research.digital.com/SRC/personal/Carl_Waldspurger/papers/register-isca93.ps

Register Communication Strategies for the Multiscalar.. - Vijaykumar Scott (1996) (Correct) (2 citations)
1 **Register** Communication Strategies for the Multiscalar
ftp.cs.wisc.edu/sohi/trs/register.1333.ps.gz

Sub-element Indexing and Probabilistic Retrieval in the POSTGRES .. - Fontaine (1995) (Correct) (1 citation)
that user can also define their own functions and **register** them with the database. POSTGRES also supports a
Sub-element **Indexing** and Probabilistic Retrieval in the POSTGRES
throughout the collection. We developed an **indexing** mechanism based on sub-element **indexing** to
wuarchive.wustl.edu/packages/postgres/papers/CSD-95-876.ps.Z

Specification and verification of the Windows Card runtime.. - Gurevich, al. (1999) (Correct)
an RTE application never leaves its "sandbox" in **scratch** memory, never learns anything about the rest of
linux.eecs.umich.edu/.5/groups/gasm/wincard.ps.gz

Google [Advanced Search](#) [Preferences](#) [Language Tools](#) [Search Tips](#)

scratch register instrument indexing | Google Search

Web · Images · Groups · Directory · News ·

Searched the web for **scratch register instrument indexing**. Results **1** - **10** of about **867**. Search took **0.14** seconds.

Instrument - Buy Musical Instruments & More on eBay Sponsored Link
www.eBay.com eBay Musical Instruments: Buy or Sell Here

Web Authoring: Software: Search for 'tracker+free+**scratch**'
Web Authoring: Software: Search for 'tracker+free+**scratch**'. ... simply by singing or playing any **instrument**. ... Advanced Keywords Conveyor 1.0 **Register** Search engines ...
webmaster.newfreeware.com/ search.php3?q=tracker%2Bfree%2Bscratch - 38k - Cached - Similar pages

NAMGAR - North American MGA **Register**
... Publishing PLC and MGA Holland **Register** Rick Astley's ... Colors - Matching Problems 23.3 Paint **Scratch** Removal 10.4 ... of Tape Deck 4.1 **Instrument** Illumination 17.4 ...
www.mgcars.org.uk/namgar/shtml/tech_sessions.shtml - 23k - Cached - Similar pages

Thomas **Register** - Industry Answers Results
Welcome to Thomas **Register** Order Online. ... Denver **Instrument** Co. ... RF Sealing, Strecth Pak, Shrinkwrap, Bagging, Card Packaging, Ink Jet, **Scratch**-Off, Cello Or Fin ...
www.thomasregister.com/ OrderOnlineCompanies.aspx?Letter=D - 87k - Cached - Similar pages

InternetDJ.com - The Global Independent Music & DJ Network
... TS404 sounds, +130 HQ wav, +800 **instrument** presets, +50 ... lovers and aspiring DJs, CD **Scratch** is the ... options (ultra-accurate pitch, mpeg **indexing**) Only personal ...
www.internetdj.com/download.php?op=MostPopular - 55k - Oct 27, 2003 - Cached - Similar pages

[PDF] BNL-Developed Sensor Shows Promise in Detecting Chemical Weapons ...
File Format: PDF/Adobe Acrobat - View as HTML
... by a telescope and analyzed by an **instrument** called a ... To **register**, pick up an application form at Human Resources ... 224/212/212/ 648 **scratch** series, M. Meier 246 ...
www.bnl.gov/bnlweb/pubaf/bulletin/1998/bb052298.pdf - Similar pages

Definitions: Index
... to the moving part of an **instrument**, intended to ... pointer, portent, presage, prognostication, **scratch**, seal, sign ... list, listing, manifest, **register**, roll, roster ...
www.websters-online-dictionary.org/ definitions/english/in/ - 101k - Cached - Similar pages

Abreviations
... Pack SP Stack Pointer SPA **Scratch** Pad Area ... Optical Recording THR Transmit Holding **Register** THX Thanks ... File Format TIGA Texas **Instrument** Graphics Architecture ...
www.lemarsu.net/abrev/s-t.html - 44k - Cached - Similar pages

Index from Faster Smarter PCs by Scott HA Clark
... timbers, 87 MP3, 99, 104 Musical **Instrument** Digital Interface ... Ctrl+Esc, 181 cursor, 9 curved **scratch** (on CD ... IEEE 1394b, 148 IEXPLORE, 277 **Indexing** Service, 70 ...
www.microsoft.com/mspress/books/index/6387.asp - 60k - Cached - Similar pages

Database Technology for Astrogrid: Status Report
... least we must encourage them to **register** their resources ... into discrete observations by each **instrument**, each of ... so the processing starts from **scratch** each time ...

www.star.le.ac.uk/~cgp/ag/dbstatus.html - 68k - Cached - Similar pages

Barriqs: an ECE 291 Final project
... Your debugging tool is a very precise **instrument**. ... General **register** usage. ... ES = **Scratch** Segment (waste of memory to swap stuff in/out of). ...
www.stg.com/employees/sbytnar/projects/ClassworkDemos_ece291/ barriqsfinal.html - 41k - Cached - Similar pages

Goooooooooogle ▶
Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

scratch register instrument indexing | Google Search | Search within results

Dissatisfied with your search results? Help us improve.

Get the Google Toolbar: Google - Search Web PageRank 6 blocked AutoFill

Yuen Chen; I-Hua Lin; Yann-Jiun Tzeng; Pi-Hsia Hung; Weichung Wang;
Computers in Education, 2002. Proceedings. International Conference on , 3-6 Dec. 2002
Page(s): 1329 -1330 vol.2

[Abstract] [PDF Full-Text (327 KB)] **IEEE CNF**

Home | Log-out | Journals | Conference Proceedings | Standards | Search by Author | Basic Search | Advanced Search
Join IEEE | Web Account | New this week | OPAC Linking Information | Your Feedback | Technical Support | Email Alerting
No Robots Please | Release Notes | IEEE Online Publications | Help | FAQ| Terms | Back to Top

Advanced Search Preferences Language Tools Search Tips

scratch register allocate indexing Google Search

Web · Images · Groups · Directory · News ·

Searched the web for **scratch register allocate indexing**. Results **21** - **30** of about **2,200**. Search took **0.12** seconds.

[PDF] THESIS **REGISTER** ALLOCATION AND ASSIGNMENT IN A RETARGETABLE ...
File Format: PDF/Adobe Acrobat - View as HTML
... In addition to a writable microcode store, secondary **scratch** memories generally exist within these micro ... The choice of which **register** to de-**allocate** is, of ...
emess.mscd.edu/~beaty/Dossier/Papers/thesis.pdf - Similar pages

By Author
... 11/15/00). Re: [PHP-WIN] Re **Indexing** of Array ... a search engine (11/08/00). [PHP-WIN] **Register** a PHP ... PHP-WIN] FATAL: emalloc(): Unable to **allocate** 1073741824 bytes ...
www.phpbuilder.com/mail/php-windows/200011/author.php - 76k - Cached - Similar pages
[More results from www.phpbuilder.com]

[PDF] Compiler and Microarchitecture Mechanisms for Exploiting ...
File Format: PDF/Adobe Acrobat - View as HTML
... I would also like to thank Steve Raasch for his help on the **register** caching work which appears as a chapter in this dissertation. ...
www.eecs.umich.edu/~tnm/theses/mattp.pdf - Similar pages

EDM/2 - An Introduction to C++ Programming - Part 13/13
... new TextPushButton(); pb->setText(txt); pb->**register**(pushed); } void ... designing a complete system from **scratch**, you can ... How to **allocate** an object of the correct ...
www.edm2.com/0608/introcpp13.html - 28k - Cached - Similar pages

afs/fs/cachefs interface.c,NONE,1.1 super.c,1.24,1.25 rootdir.c, ...
... rec.**scratch**); free_page((unsigned long) rec.**scratch**); - kleave(" = %d ... 432,22 +438,20 @@ /* * **allocate** an entry ... goto error; + + printk("\n### **Register** file object ...
lists.infradead.org/pipermail/linux-afs-cvs/ 2003-April/000006.html - 78k - Cached - Similar pages

[PDF] Microsoft PowerPoint - Lecture-Mar-6
File Format: PDF/Adobe Acrobat - View as HTML
... low level • Examples – GCC's RTL (**Register** Transfer Language ... minus, logical not • Array **indexing** (bounds check ... Read a line _Readline **Allocate** memory int ...
www.cs.drexel.edu/~souter/cs761/ lectures/lecture9/lecture9.pdf - Similar pages

<html> <head> </head><body><pre>/* Subroutines for insn-output.c ...
... inline version to be slightly longer in some cases as it saves a **register**. ... want - we generate the code at emit time, but need to **allocate** a **scratch** reg now ...
www.mit.edu/afs/athena/astaff/project/ gccdev/src/config/mn10200/mn10200.c - 44k - Cached - Similar pages

<html> <head> </head><body><pre>This is Info file gcc.info, ...
... in smaller code, but slower execution, since **scratch** space must ... REGS is a series of letters specifying a **register**. ... D&#39; **allocate** EDI; `B&#39; **allocate** EBP. ...
www.mit.edu/afs/sipb/project/egcs/info/gcc.info-5 - 45k - Cached - Similar pages
[More results from www.mit.edu]

PART II. THE FORTH KERNEL
... DI **Scratch**. ES Extra segment. ... F83 **allocate** 4 Kbytes for the buffers, enough to hold 4 blocks of data from/to the disk. ... AX AX OR Set the CPU status **register**. ...

www.eforth.com.tw/academy/library/ insidef83%5Chostinterface.htm - 101k - Cached - Similar pages

[PDF] Programming Systolic Arrays
File Format: PDF/Adobe Acrobat - View as HTML
... alternately used for communication The use of a local **scratch register** could solve ... store step is to detect these hazards and **allocate** registers appropriately ...
www.cse.ucsc.edu/research/kestrel/papers/asap92.pdf - Similar pages

◀ Gooooooooooogle ▶
Result Page: **Previous** 1 2 3 4 5 6 7 8 9 10 11 12 **Next**

scratch register allocate indexing | Google Search | Search within results

Google Home - Advertise with Us - Business Solutions - Services & Tools - Jobs, Press, & Help

IEEE HOME | SEARCH IEEE | SHOP | WEB ACCOUNT | CONTACT IEEE

IEEE

Membership Publications/Services Standards Conferences Careers/Jobs

IEEE Xplore® RELEASE 1.5

Welcome
United States Patent and Trademark Office

Help FAQ Terms IEEE Quick Links » Search Results
Peer Review

Welcome to IEEE Xplore®
- Home
- What Can I Access?
- Log-out

Tables of Contents
- Journals & Magazines
- Conference Proceedings
- Standards

Search
- By Author
- Basic
- Advanced

Member Services
- Join IEEE
- Establish IEEE Web Account
- Access the IEEE Member Digital Library

Print Format

Your search matched **4** of **981130** documents.

A maximum of **4** results are displayed, **15** to a page, sorted by **Relevance** in **descending** order.
You may refine your search by editing the current search expression or entering a new one the text box.
Then click **Search Again**.

(register)and (scratch)

Search Again

Results:
Journal or Magazine = **JNL** Conference = **CNF** Standard = **STD**

1 **On the problem of in-place calculating LPC parameters**
Xixian Chen; Weixin Li;
Circuits and Systems, 1991., IEEE International Sympoisum on , 11-14 June 1991
Page(s): 316 -319 vol.1

[Abstract] [PDF Full-Text (244 KB)] **IEEE CNF**

2 **Conflict-free access to multiple single-ported register files**
Mueller, S.M.; Vishkin, U.;
Parallel Processing Symposium, 1997. Proceedings., 11th International , 1-5 April 1997
Page(s): 672 -678

[Abstract] [PDF Full-Text (560 KB)] **IEEE CNF**

3 **Reverse compilation of digital signal processor assembler source to ANSI-C**
Johnstone, A.; Scott, E.; Womack, T.;
Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on , 30 Aug.-3 Sept. 1999
Page(s): 316 -325

[Abstract] [PDF Full-Text (184 KB)] **IEEE CNF**

4 **Assessment design for mathematics web project-based learning**
[illegible]

First Experiences with a System for Content Based Retrieval .. - Schäuble, Wechsler (1995) (Correct) (3 citations)
and accepts textual queries. The fully automatic **indexing** was done using speech recognition techniques.
was done using speech recognition techniques. **Indexing** speech documents is challenging, because word
errors influence the retrieval effectiveness. Our **indexing** vocabulary consists of 5000 phone sequences. To
ftp.inf.ethz.ch/doc/papers/is/ir/ijcai95.ps.gz

Optimum Modulo Schedules for Minimum Register Requirements - Eichenberger, Davidson.. (1995) (Correct) (12 citations)
Optimum Modulo Schedules for Minimum **Register** Requirements Alexandre E. Eichenberger and
www4.ncsu.edu/~alexe/papers/ICS95.ps

Resource Spackling: A Framework for Integrating Register.. - Berson, Gupta, Soffa (1994) (Correct) (16 citations)
In the special case where a **register** is used as **scratch** space during the execution of a complex
Resource Spackling: A Framework for Integrating **Register** Allocation in Local and Global Schedulers yz
units in a single phase so that all resources are **allocate**d at once. Our approach is able to consider the
ftp.cs.pitt.edu/berson/papers/tr94-09.ps

Loop Scheduling Algorithm for Timing and Memory Operation.. - Fei Chen (Correct)
for Timing and Memory Operation Minimization with **Register** Constraint Fei Chen Sissades Tongsima Edwin
www.nd.edu/~esha/papers/fei/SiPS98.ps

Optimistic Register Coalescing - Park, Moon (1998) (Correct)
after which the allocation steps are repeated from **scratch**. The phase ordering of the Chaitin's coloring
Optimistic **Register** Coalescing Jinpyo Park Soo-Mook Moon School of
since a shorter live range is more likely to be **allocated registers** than a longer one. The cost involved
deneb.snu.ac.kr/pub/jp/publications/jp-pact98.ps

Complexity of Finding Alphabet Indexing - Shimozono, Miyano (1995) (Correct)
Complexity of Finding Alphabet **Indexing** Shinichi Shimozono Department of Control
and Q of strings over an alphabet 6, an alphabet **indexing** /for P Q by an **indexing** alphabet 0 with j0j !
alphabet 6, an alphabet **indexing** /for P Q by an **indexing** alphabet 0 with j0j !j6j is a mapping /6 !0
www.i.kyushu-u.ac.jp/TR/61.ps.Z

A Connectonist Indexing Approach for CBR Systems - Malek (Correct)
defined as the locus of points in the space that **register** a measure of similarity equal or greater than a
A Connectonist **Indexing** Approach for CBR Systems Maria Malek
we show how we can construct a simple efficient **indexing** system structure. We construct a case hierarchy
cosmos.imag.fr/RESEAUX/malek/malek.iccbr95.ps.gz

SPLITDAT and DECOMP: Two New GAMS I/O Subroutines to.. - Chang, Fragnière (1996) (Correct)
This communication is handled through a set of **scratch** files generated by GAMS (scr)The
The master problem makes decisions on how to **allocate** scarce resources in the first stage. The
describe the matrix. The storage technique is "row **index**, column pointer"i.e. column wise. The variables
ecolu-info.unige.ch/%7Elogilab/reports/mleval.ps

First 20 documents Next 20

Try your query at: Amazon Barnes & Noble Google (RI) Google (Web) CSB DBLP